02｜蓄势待发：秒杀系统架构设计和环境准备

2021-09-27 佘志东

《手把手带你搭建秒杀系统》 课程介绍 >

讲述：佘志东
时长 12:51 大小 11.77M

你好，我是志东，欢迎和我一起从零打造秒杀系统。

我们知道，系统的设计是个由巨入细的过程，想去设计好它，那你首先得去了解清楚它。就像上节课我们对 HTTP 请求所走链路的介绍，学完后你就会明白，做秒杀系统设计时，会用到哪些层级系统，并且每个层级系统可以做什么事情。

今天我们要做的就是给每个层级系统做最合适的**技术选型和职能边界划分**，最终实现让各系统、技术做它们所擅长的事情，并在最后搭建起我们的开发依赖环境。

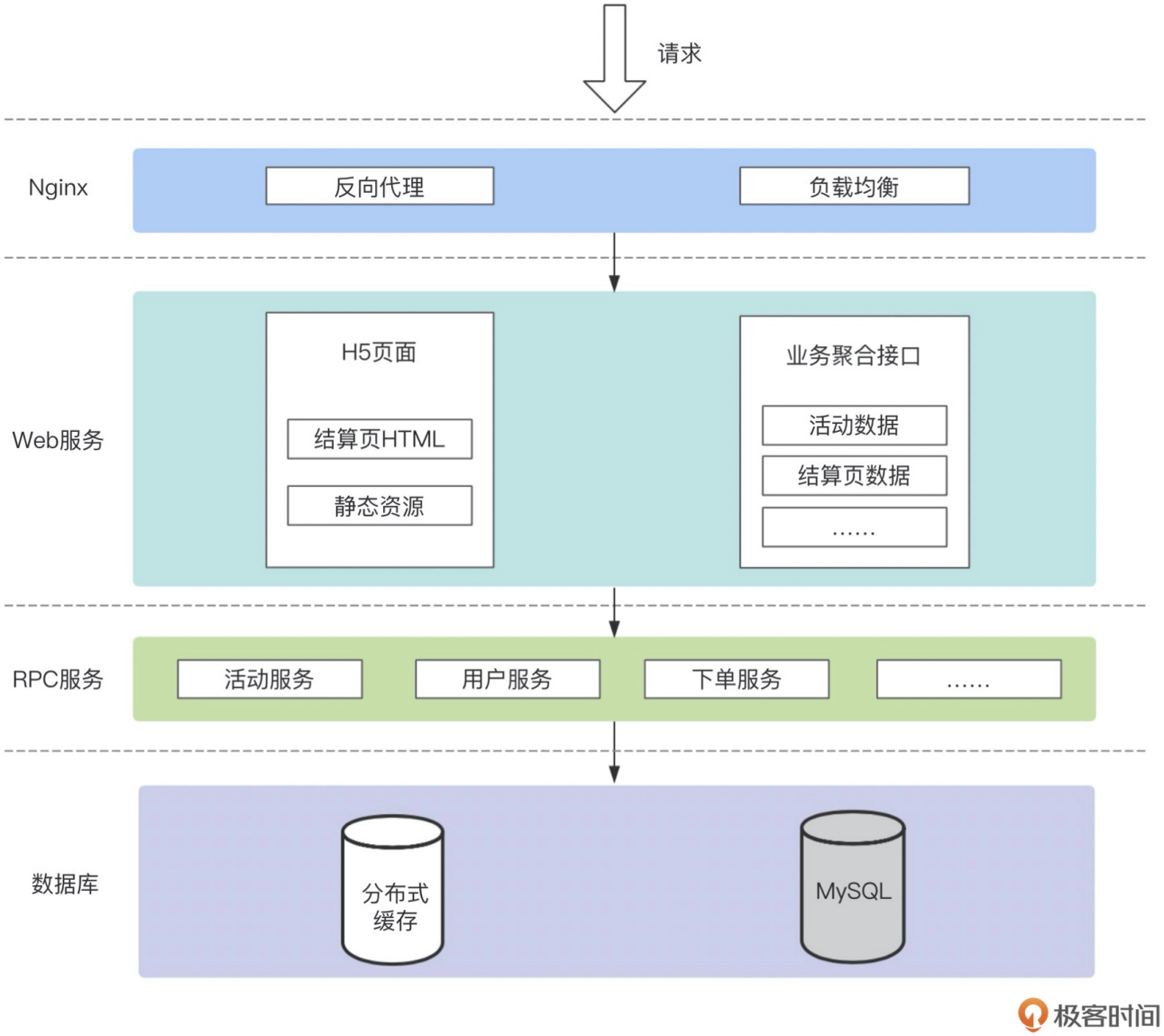
那如何给层级系统做技术选型和职能边界划分呢？我们通常都说，没有最好的技术，只有最契合当下业务场景的技术，所以我们得先了解一下，如果使用我们传统的架构系统来支

持秒杀业务，可能会出现哪些问题。只有清楚了要面对的问题，我们才能做针对性的思考和优化。

所以这节课我们将重点分析传统架构设计的特点，接着介绍最新的秒杀系统架构，并做好技术选型和环境准备。

传统秒杀系统架构

下面先看一个大家常用的系统功能架构图：



这种功能结构以及系统架构，是我们非常熟悉的。在这种方式下，Nginx 只做反向代理和负载均衡，甚至这层对我们做业务开发的研发人员来说，都是无感知的，一般运维同事在做生产环境搭建时，都会帮我们配好。研发人员更多的是在开发 Web 服务和 RPC 服务，

我们把页面以及页面所依赖的静态资源都放到 Web 服务中，同时 Web 服务还提供业务接口，RPC 服务提供一些支撑服务。

如果这是个 ToB 的运营管理系统，这样没有什么问题，因为请求量非常低，系统基本不会有太大的负载。但是对于 ToC，且瞬时流量非常大的情况，问题就会暴露出来，那它究竟会有哪些问题呢？

域名与带宽问题

我们从最基础的讲起。如果 Web 服务既提供 H5 页面、静态资源，同时也提供业务接口，这就意味着所有的请求使用的都是同一个域名，在活动刚开始时，大家都点击抢购按钮进结算页，而结算页页面拉取静态资源，会占用很多带宽资源。

这在活动开始的瞬间，带宽资源很稀缺的情况下，可能会出现用户进不了结算页，或者进了结算页却不能正常渲染页面的问题，导致抢购体验大幅下降。

Web 服务器性能问题

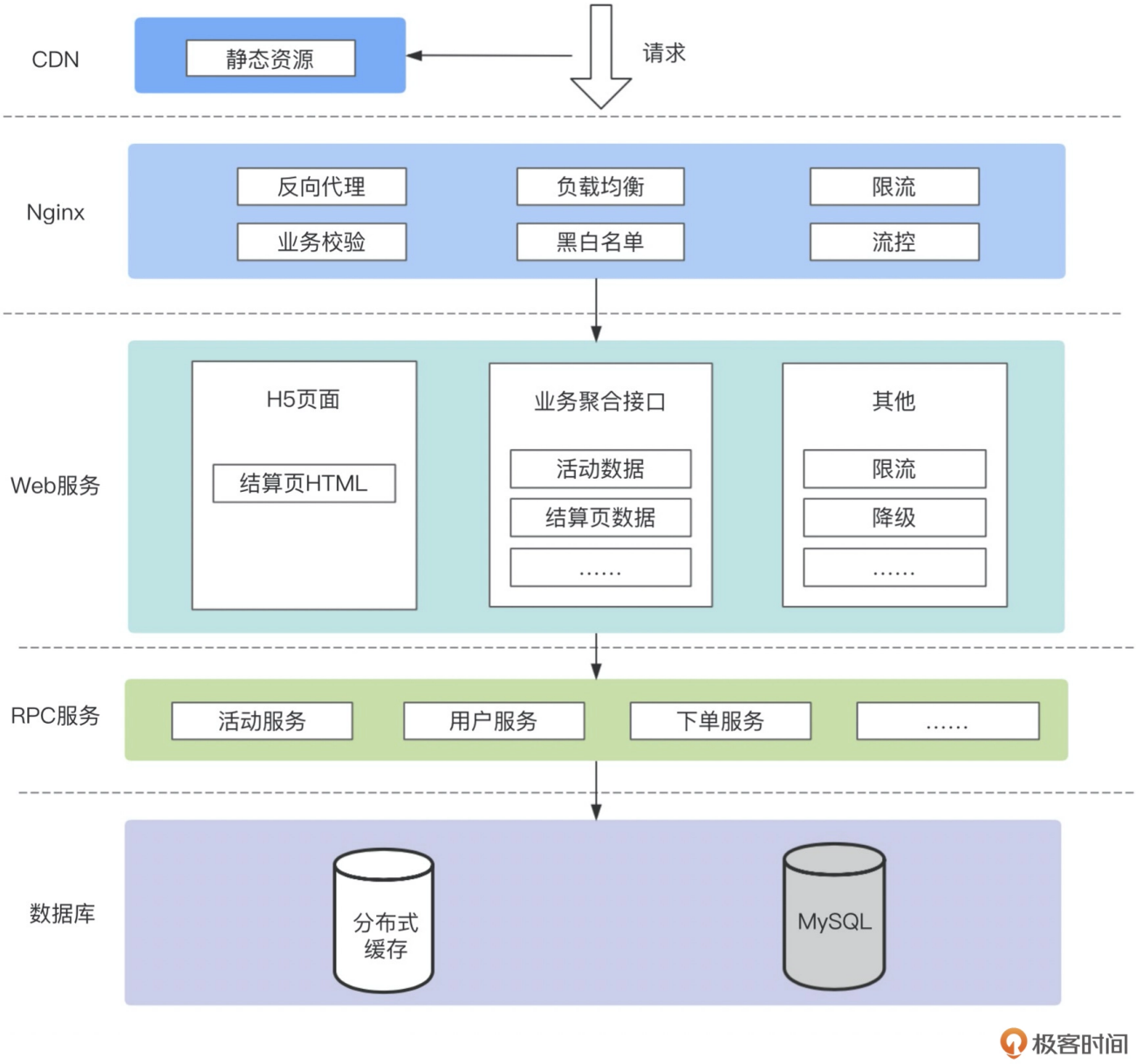
接着，讲一个关键问题。我们一般部署 Web 服务，都是使用 Apache 的 Tomcat 来部署的，Tomcat 在处理请求的时候，是通过线程去处理的。

这样的问题就是如果瞬时的大量请求过来，线程池中的线程不够用，Tomcat 就会瞬间新建很多线程，直至达到配置的最大线程数，如果线程数设置的过大，这个过程可能会直接将机器的 CPU 打满，导致机器死掉。即使没有挂掉，在高负载下，当设置的等待队列也满了之后，后面的请求都会被拒绝连接，直到有空出的资源去处理新请求。这时候你可能会想，我加机器分摊流量不就行了？可以是可以，但由此增加的活动成本不知道你的老板会不会买单？

当然了，这个过程中，还会伴有热点数据读写、库存超卖等问题，这些细节也非常重要，我会在后面的课程中一一给你展开说明。

新的秒杀系统架构

上面我们谈到了传统系统架构的局限性，那么我们新的系统该如何设计才能避免出现以上问题呢？结合上节课对于各链路层级的介绍，我画了一张新的功能结构与系统架构图：



首先新系统我们依然保留了 HTTP 服务常用的层级调用关系，即 **Nginx->Web 服务->RPC 服务**，这也是绝大部分公司都会使用的一种系统结构。

其次将原先由 Web 服务提供的静态资源放到了 CDN（CDN 是全国都有的服务器，客户端可以根据所处位置自动就近从 CDN 上拉取静态资源，速度更快），来大大减轻抢购瞬时秒杀域名的负担。

最后，同时也是我们所做的最大改变，就是**将 Nginx 的职责放大**，前置用来做 Web 网关，承担部分业务逻辑校验，并且增加黑白名单、限流和流控的功能，这其实也是考虑到我们的秒杀业务特点所做的调整。这种在 Nginx 里写业务的做法在很多大公司里都是很常见的，像京东是用来做商详、秒杀的业务网关，美团用来做负载均衡接入层，12306 用来

做车票查询等等，他们的共同特点都是要面对高并发的业务场景，这也说明在这种业务场景下，我们的设计是得到了真实实践和广泛认可的。

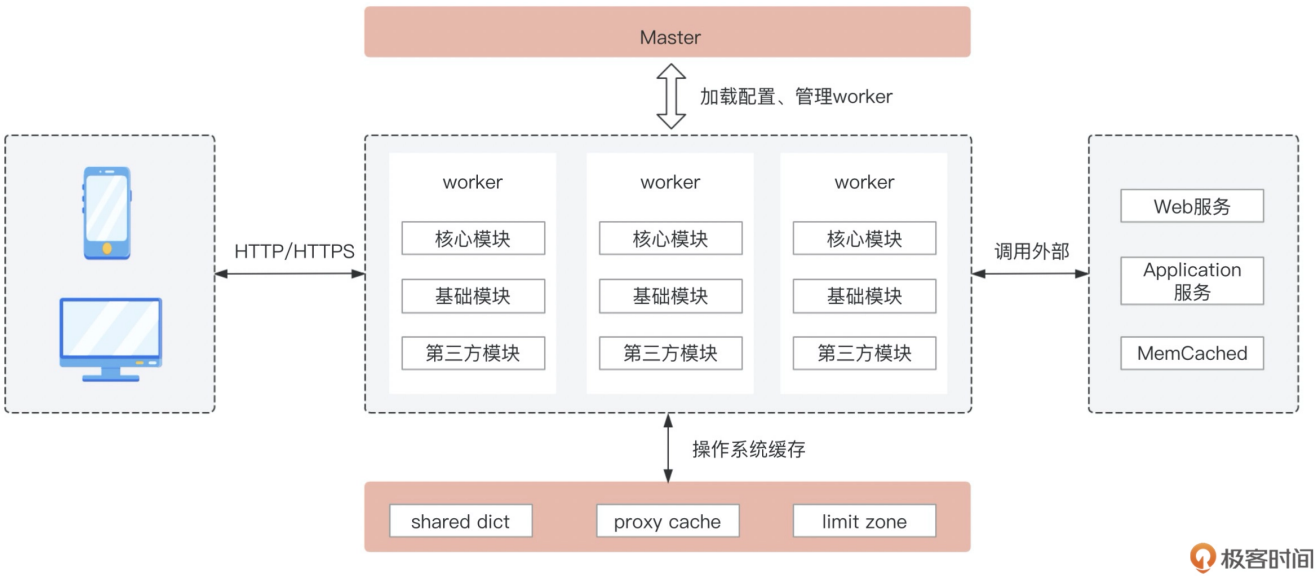
而这么做的目的，就是要充分利用 Nginx 的高并发、高吞吐能力，并且非常契合我们秒杀业务的特点，即入口流量大。但流量组成却非常的混杂，这些请求中，一部分是刷子请求，一部分是无效请求（传参等异常），剩下的才是正常请求，这个的比例可能是 6：1：3，所以需要我们在网关层尽可能多地接收流量进来，并做精确地筛选，将真正有效的 3 成请求分发到下游，剩余的 7 成拦截在网关层。不然把这些流量都打到 Web 服务层，Web 服务再新起线程来处理刷子和无效请求，这是种资源的浪费。

所以网关层对秒杀系统而言，至关重要，而 Nginx 刚好可以胜任此项任务。由此可见，Nginx 在我们的系统设计中，扮演着非常重要的角色，但你对 Nginx 也许没那么了解，别急，接下来我就给你简单介绍一下 Nginx，并带你解开 Nginx 在高并发下仍具有高性能的秘密。

Nginx 介绍

Nginx 最早被发明出来，就是来应对互联网高速发展下，出现的并发几十万、上百万的网络请求连接场景的，传统 Apache 服务器无法有效地解决这种问题，而 Nginx 却具有并发能力强、资源消耗低的特性。

总的来说，Nginx 有 5 大优点，即**模块化、事件驱动、异步、非阻塞、多进程单线程**。以下是 Nginx 的架构原理图：



Nginx 是由一个 master 进程和多个 worker 进程（可配置）来配合完成工作的。其中 master 进程负责 Nginx 配置文件的加载和 worker 进程的管理工作，而 worker 进程负责请求的处理与转发，进程之间相互隔离，互不干扰。同时每个进程中只有一个线程，这就省去了并发情况下的加锁以及线程的切换带来的性能损耗。

但一个线程能支持高并发的业务场景吗？

这就要说到 Nginx 的工作模型。以 Linux 为例，其采用的是 epoll 模型（即事件驱动模型），该模型是 IO 多路复用思想的一种实现方式，是异步非阻塞的，什么意思呢？就是一个请求进来后，会由一个 worker 进程去处理，当程序代码执行到 IO 时，比如调用外部服务或是通过 upstream 分发请求到后端 Web 服务时，IO 是阻塞的，但是 worker 进程不会一直在这等着，而是等 IO 有结果了再处理，在这期间它会去处理别的请求，这样就可以充分利用 CPU 资源去处理多个请求了。

这里你还可以思考这样一个问题：Linux 支持的以 IO 多路复用思想来实现的模型还有 select 和 poll，为什么选择了 epoll 呢？**因为 epoll 的效率更高。**

举个例子，刚刚我们上面说到 worker 在处理请求到 IO 时，不会阻塞等待，而是去干其他事情，等 IO 有结果了再回头处理，那 worker 进程怎么知道刚刚的 IO 处理完毕了呢？

假设一个 work process 处理了 1000 个连接，但其中只有 10 个 IO 完成了，并可以继续往下执行，select/poll 的做法是遍历这 1000 个 FD（File Description，可以理解成每个建立了连接的一个标识），找到那 10 个就绪状态的，并把没做完的事情继续做完，这样检索的效率明显很低。所以 epoll 的做法是当这 10 个 IO 准备就绪时，通过系统的回调函数将 FD 放到一个专门的就绪列表中，这样系统只需要去找这个就绪列表就可以了，这大大提高了系统的响应效率。当然这只是 epoll 的其中一个优点，具体三种模型的对比，你可以自行去了解一下，网上相关的资料有很多，或者我们在留言区讨论也是可以的。

正是**多进程 + 事件驱动**的工作原理，才使得 Nginx 具有非常良好的性能表现，同时 Nginx 的模块化，也能够支撑强大的第三方自定义工具模块，让你的开发更加灵活自由。

OpenResty 介绍

我们知道，Nginx 的底层模块一般都是用 C 语言写的，如果我们想在 Nginx 的基础之上写业务逻辑，还得借助 OpenResty，它是 Nginx 的一个社区分支。这里我也简单介绍一

下它。

OpenResty 是一个基于 Nginx 与 Lua 的高性能 Web 平台，它使我们具备在 Nginx 上使用 Lua 语言来开发业务逻辑的能力，并充分利用 Nginx 的非阻塞 IO 模型，来帮助我们非常方便地搭建能够处理超高并发、扩展性极高的动态 Web 应用、Web 服务和动态网关。

这里插一句，**为什么要用 Lua 语言来做 Nginx 开发呢？**这就要说到 Lua 语言的特点了，Lua 的线程模型是单线程多协程的模式，而 Nginx 刚好是单进程单线程，天生的完美搭档。同时 Lua 是一种小巧的脚本语言，语法非常的简单，很容易学习掌握，所以对于新语言你先不要有排斥心理，我会在后面的课程中慢慢向你展示讲解。

Web/RPC 服务技术选型

以上我介绍完了 Nginx 服务层的技术选型，同时也讲解了为什么这么选的原因，下面就轮到 Web 服务和 RPC 服务了。

这里大的框架选择，其实就没有太多要求了，只要能提供我们需要的能力即可，比如基础框架是使用 SpringMVC 还是 SpringBoot，持久层是喜欢用 MyBatis 还是 JPA，数据库是用 MySQL 还是 Oracle，这些都可以根据你的个人使用习惯或者所在公司的技术栈做灵活变通。

同时对于已经有秒杀系统，但是想要做优化的情况，也完全不用担心，跟着我学习之后，你只需将旧系统中的部分轻业务逻辑迁移到 Nginx 层来，体量最大的业务逻辑代码基本都不用动的，并且旧系统中的一些优化点，也都有单独的技术来实现，而这些都不需要太多的学习成本和迁移成本。

那么为了更好的本地开发教学，这里介绍一下**我所使用的技术栈**：Web 服务和 RPC 服务的基础框架都是使用 SpringMVC，RPC 框架使用的是 Dubbo，数据库使用免费开源的 MySQL，分布式缓存数据库使用 Redis，这应该也是大多数公司会使用的技术栈。

在技术选型和层级系统职能划分都确定了之后，接下来就让我们动起手来，先把开发的依赖环境准备好。

环境准备

以下是我在本地 Mac 上的安装方式，如果你是其他系统，可以找对应的安装方法，对于程序员来说应该是没难度的。如果 Mac 上没有安装过 Homebrew，可以安装一下，这是一个 Mac 软件的工具包，很好用。以下操作都是在 Mac 的终端里输入相关命令来完成操作的。

Homebrew 安装

ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

OpenResty 安装

1. brew install openresty/brew/openresty

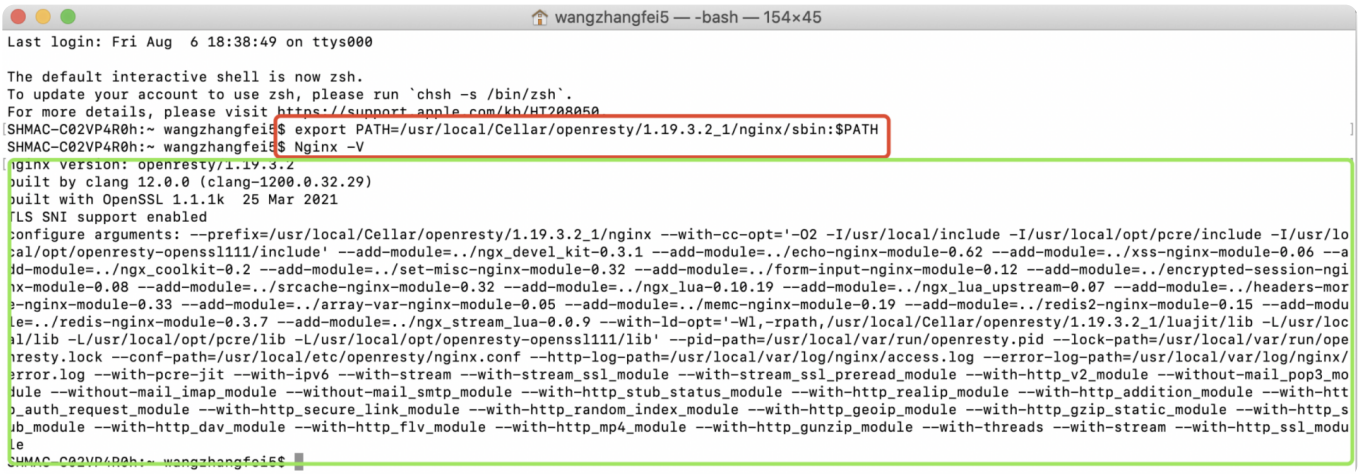
从 Homebrew 安装 OpenResty。

2. export PATH=/usr/local/Cellar/openresty/1.19.3.2_1/nginx/sbin:$PATH

安装完成之后，默认的安装位置在 /usr/local/Cellar/openresty/1.19.3.2_1，这时我们设置下环境变量（即告诉终端输入的命令去哪里找）。

3. Nginx -V

然后就可以查看 OpenResty 是否安装成功。如下图，执行红框内的命令，出现绿框的输出内容，即表示安装成功了。


wangzhangfei5 — -bash — 154×45
Last login: Fri Aug 6 18:38:49 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050
SHMAC-C02VP4R0h:~ wangzhangfei5$ export PATH=/usr/local/Cellar/openresty/1.19.3.2_1/nginx/sbin:$PATH
SHMAC-C02VP4R0h:~ wangzhangfei5$ Nginx -V
nginx version: openresty/1.19.3.2
built by clang 12.0.0 (clang-1200.0.32.29)
built with OpenSSL 1.1.1k 25 Mar 2021
TLS SNI support enabled
configure arguments: --prefix=/usr/local/Cellar/openresty/1.19.3.2_1/nginx --with-cc-opt='-O2 -I/usr/local/include -I/usr/local/opt/pcre/include -I/usr/lo
cal/opt/openresty-openssl111/include' --add-module=../ngx_devel_kit-0.3.1 --add-module=../echo-nginx-module-0.62 --add-module=../xss-nginx-module-0.06 --a
dd-module=../ngx_coolkit-0.2 --add-module=../set-misc-nginx-module-0.32 --add-module=../form-input-nginx-module-0.12 --add-module=../encrypted-session-ngi
nx-module-0.08 --add-module=../srcache-nginx-module-0.32 --add-module=../ngx_lua-0.10.19 --add-module=../ngx_lua_upstream-0.07 --add-module=../headers-mor
e-nginx-module-0.33 --add-module=../array-var-nginx-module-0.05 --add-module=../memc-nginx-module-0.19 --add-module=../redis2-nginx-module-0.15 --add-modu
le=../redis-nginx-module-0.3.7 --add-module=../ngx_stream_lua-0.0.9 --with-ld-opt='-Wl,-rpath,/usr/local/Cellar/openresty/1.19.3.2_1/luajit/lib -L/usr/loc
al/lib -L/usr/local/opt/pcre/lib -L/usr/local/opt/openresty-openssl111/lib' --pid-path=/usr/local/var/run/openresty.pid --lock-path=/usr/local/var/run/ope
nresty.lock --conf-path=/usr/local/etc/openresty/nginx.conf --http-log-path=/usr/local/var/log/nginx/access.log --error-log-path=/usr/local/var/log/nginx/
error.log --with-pcre-jit --with-ipv6 --with-stream --with-stream_ssl_module --with-stream_ssl_preread_module --with-http_v2_module --without-mail_pop3_mo
dule --without-mail_imap_module --without-mail_smtp_module --with-http_stub_status_module --with-http_realip_module --with-http_addition_module --with-htt
p_auth_request_module --with-http_secure_link_module --with-http_random_index_module --with-http_geoip_module --with-http_gzip_static_module --with-http_s
ub_module --with-http_dav_module --with-http_flv_module --with-http_mp4_module --with-http_gunzip_module --with-threads --with-stream --with-http_ssl_modu
le
SHMAC-C02VP4R0h:~ wangzhangfei5$

4. 然后我们就可以测试下 Nginx 是否好用，所以我们在本地新建了个 nginx.conf 配置文件，就放在 /Users/~/Documents/seckillproject/nginx/conf 下，并且在 nginx 文件夹下新建 logs 文件夹，用于 log 日志的输出，新建后的文件目录结构如下：

nginx.conf 的内容就用官方的模板，输出个 hello world：

复制代码

```
worker_processes  1;
error_log logs/error.log;
events {
    worker_connections 1024;
}
http {
    server {
        listen 8080;
        location / {
            default_type text/html;
            content_by_lua_block {
                ngx.say("<p>hello, world</p>")
            }
        }
    }
}
```

5. cd /Users/wangzhangfei5/Documents/seckillproject/nginx

进入到我们新建的 nginx 文件夹下。

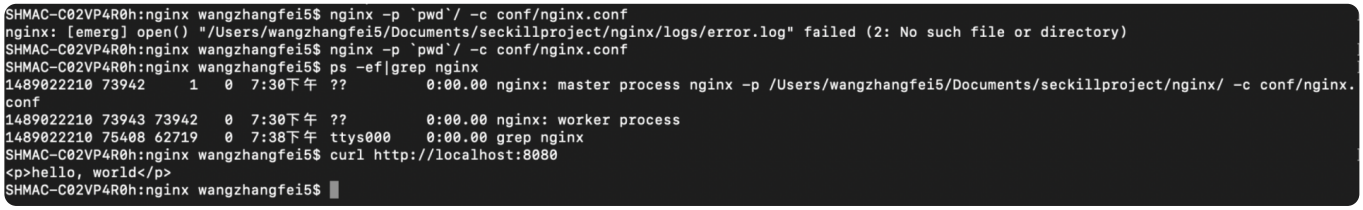
6. nginx -p `pwd`/ -c conf/nginx.conf

启动 Nginx 服务，这时输入 `ps -ef|grep nginx` 可以查看起来 Nginx 进程，有两个，一个 master ，一个 worker。

SHMAC-C02VP4R0h:nginx wangzhangfei5$ nginx -p `pwd`/ -c conf/nginx.conf
SHMAC-C02VP4R0h:nginx wangzhangfei5$ ps -ef|grep nginx
1489022210 73942 1 0 7:30下午 ?? 0:00.00 nginx: master process nginx -p /Users/wangzhangfei5/Documents/seckillproject/nginx/ -c conf/nginx.
conf
1489022210 73943 73942 0 7:30下午 ?? 0:00.00 nginx: worker process
1489022210 75408 62719 0 7:38下午 ttys000 0:00.00 grep nginx
SHMAC-C02VP4R0h:nginx wangzhangfei5$

7. curl http://localhost:8080

访问本地 8080 端口，可以看到，输出了" hello, world "，也可以在浏览器输入 http://localhost:8080，看到 hello,world 的返回。


SHMAC-C02VP4R0h:nginx wangzhangfei5$ nginx -p `pwd`/ -c conf/nginx.conf
nginx: [emerg] open() "/Users/wangzhangfei5/Documents/seckillproject/nginx/logs/error.log" failed (2: No such file or directory)
SHMAC-C02VP4R0h:nginx wangzhangfei5$ nginx -p `pwd`/ -c conf/nginx.conf
SHMAC-C02VP4R0h:nginx wangzhangfei5$ ps -ef|grep nginx
1489022210 73942 1 0 7:30下午 ?? 0:00.00 nginx: master process nginx -p /Users/wangzhangfei5/Documents/seckillproject/nginx/ -c conf/nginx.
conf
1489022210 73943 73942 0 7:30下午 ?? 0:00.00 nginx: worker process
1489022210 75408 62719 0 7:38下午 ttys000 0:00.00 grep nginx
SHMAC-C02VP4R0h:nginx wangzhangfei5$ curl http://localhost:8080
hello, world
SHMAC-C02VP4R0h:nginx wangzhangfei5$

8. nginx -p `pwd` -s stop

停止 Nginx 服务。

MySQL 安装

brew install mysql@5.7

也可以通过官网下载，根据系统版本，下载好对应包之后直接安装，完成安装之后可以测试下。

1. export PATH=/usr/local/opt/mysql@5.7/bin:$PATH

默认的安装位置在 /usr/local/opt/mysql@5.7 这时我们设置下环境变量（即告诉终端输入的命令去哪里找）。

2. mysql.server start

启动 MySQL。

3. mysql_secure_installation

设置数据库密码，按照对应的提示，让选择 Y/N 时，输入 Y，然后会让选择密码等级，一共三个级别 0,1,2 强度由低到强，选择后设置密码，并记住刚设的密码。

4. mysql -uroot -p

登录数据库，这时会提示你输入密码，输入刚设置的密码即可进入。

5. show databases;

查看下当前的所有库，到这里也说明我们的 MySQL 准备好了，可以建库建表了。

```
All done!
SHMAC-C02VP4R0h:~ wangzhangfei5$ mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.7.35 Homebrew

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql>
```

6. mysql.server stop

control+z 退出 MySQL 后，执行该命令，关闭 MySQL。

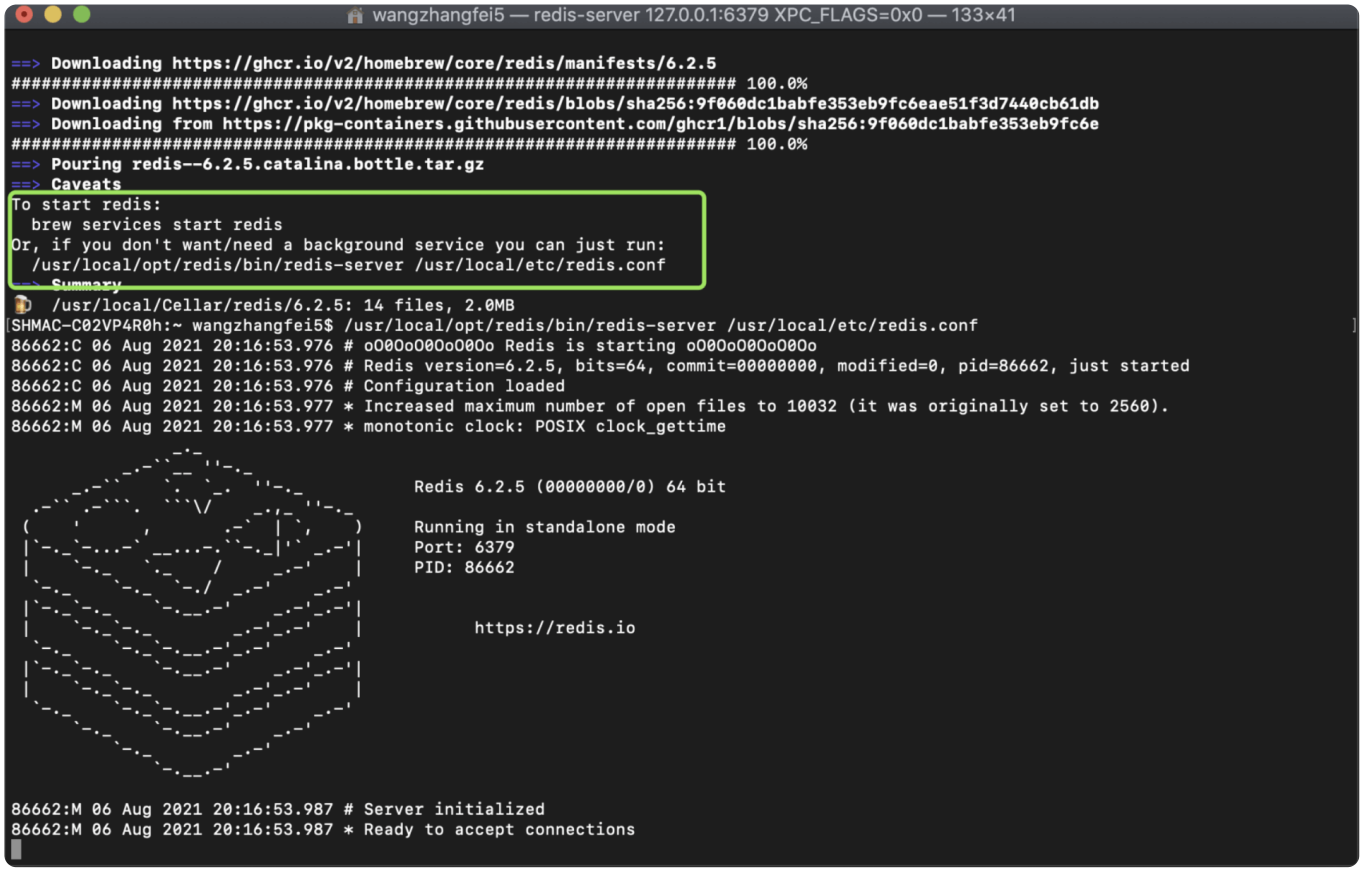
Redis 安装

brew install redis

安装完成之后 ，依次执行以下命令进行测试。

1. /usr/local/opt/redis/bin/redis-server /usr/local/etc/redis.conf

启动 Redis 服务端，如下图绿框，在安装好后提示我们如何启动 Redis，按照提示输入命令，便可以看到启动成功等待连接。



2. redis-cli -h 127.0.0.1 -p 6379

这时新建一个终端窗口，模拟客户端连接 Redis 服务，如下图所示，可以 set 一个值，并 get 查询出来，说明 Redis 也正常安装成功并可以使用。

3. 服务端停止，可以直接 control+z 退出，也可以 sudo pkill redis-server 客户端断开连接 redis-cli shutdown。

到这我们主要的依赖环境都已经搭好了，而项目的搭建与开发，我会在下节课为你讲解。

小结

在这节课里，针对秒杀系统，我们将传统的架构设计与我们新的架构设计做了一个对比，可以看出传统架构设计的局限性，其中仅列举了域名带宽问题和 Tomcat 服务器性能问题，这也是我们从宏观上做技术选型时，就需要去认真思考的问题。

而针对这两点，我们也给出了我们的答案，即利用 Nginx 在高并发下仍具有高性能的特性，将 Web 网关职能前置，尽量在流量入口处拦截掉风险流量以及缩短请求链路，保护下游系统，并提高服务的响应速度。

有了大的方向指导，我们便可以针对一些更细的技术点去做优化和设计，比如哪种限流算法更好，怎么能在高并发下保证库存不超卖等等这些。

同时在做技术选型时，我也尽可能地使用了多数公司都在使用的技术栈，以降低你的学习成本。但是用 Lua 语言来做 Nginx 业务开发，或许还有不少同学是第一次用，不过不着急，后面跟着我慢慢学，相信你自己一定可以，或许这还将成为你比别人厉害的法宝！

思考题

关于 Tomcat 的思考，为什么 Tomcat 也支持 NIO，但性能却比 Nginx 差那么多呢？

以上就是这节课的全部内容，欢迎你在评论区和我讨论问题，交流想法！

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 4 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 01 | 直面痛点：秒杀系统的挑战和设计原则

下一篇 03 | 指日可待：一步一步搭建秒杀系统（上）

精选留言 (6)

写留言

Z.G
2021-09-30

老师你好，问下如果是基于现代云原生的K8s基础设施，还需要Nginx吗？又应该怎么处理呢？

1

黄序
2021-10-01

1）两者的使用侧重点不同，Nginx主要是反向代理以及负载均衡，Tomcat是一个servlet容器，需要处理各种动态请求，比如说网络IO以及和数据库的交互；
2）机制不同：作者大大提及到，lua脚本底层用到了携程，Tomcat使用的是线程，性能上也会存在差距

展开

Geek_5b2ab1
2021-09-30

踩坑之一： nginx -p pwd/ -c conf/nginx.conf 应为nginx -p ·pwd·/ -c conf/nginx.conf pwd左右的是反引号，表示执行pwd命令，获取当前工作目录路径

作者回复: 坑已填平，``包裹没有正常显示。

nana
2021-09-28

麻烦速更啊啊啊，完全不够看

送过快递的码农
2021-09-27

我觉得可能是tomcat 和 Nginx的侧重点是不一样，nginx讲究主要管这个请求，从哪儿来，到哪儿去。它不求解决所有问题，是一个专业的中转站。而tomcat是一个servlet容器，它讲究怎么吃下这些请求，并且都处理好，因此，它需要适配各种消息体，处理不同的五花八门的请求，甚至于还得考虑，cookie，session，上下文等问题。虽然tomcat很伟大的用servlet和servlet容器的方式来解耦各个业务请求，减少了开发成本，但是性能成本确实挥之不去了

1

范

2021-09-27

对于Windows系统，对应的软件是什么？mysql+redis+nginx+IIS?

2